Gaussian Process-based Min-norm Stabilizing Controller for Control-Affine Systems with Uncertain Input Effects and Dynamics

Fernando Castañeda*, Jason J. Choi*, Bike Zhang, Claire J. Tomlin and Koushil Sreenath

***Abstract*— This paper presents a method to design a min-norm Control Lyapunov Function (CLF)-based stabilizing controller for a control-affine system with uncertain dynamics using Gaussian Process (GP) regression. In order to estimate both state and input-dependent model uncertainty, we propose a novel compound kernel that captures the control-affine nature of the problem. Furthermore, by the use of GP Upper Confidence Bound analysis, we provide probabilistic bounds of the regression error, leading to the formulation of a CLF-based stability chance constraint which can be incorporated in a min-norm optimization problem. We show that this resulting optimization problem is convex, and we call it "Gaussian Process-based Control Lyapunov Function Second-Order Cone Program" (GP-CLF-SOCP). The data-collection process and the training of the GP regression model are carried out in an episodic learning fashion. We validate the proposed algorithm and controller in numerical simulations of an inverted pendulum and a kinematic bicycle model, resulting in stable trajectories which are very similar to the ones obtained if we actually knew the true plant dynamics.**

I. Introduction

Model-based controllers have a problem inherent to their nature: model uncertainty. In this paper, we directly address this issue for the case of Lyapunov-based stabilizing controllers for nonlinear control-affine systems by using Gaussian Process (GP) regression to estimate the adverse effects of model uncertainty.

Control Lyapunov Functions (CLFs) [1], [2] have been widely used in recent years for nonlinear model-based stabilizing control of robotic systems [3], [4], [5]. Typically, the robot is stabilized by enforcing the CLF to decay to zero with a constraint in an optimization problem [6]. However, CLF-based optimization methods heavily rely on the assumption that the model used for the controller design accurately represents the true plant's dynamics. If there is model-plant mismatch, convergence guarantees are often lost. Past research has directly addressed this issue by using both robust [4] and adaptive [7] control theory. More recently, various kinds of data-driven methods that use neural networks have been introduced [8], [9], [10]. Although these are demonstrated to be effective in practice, it is often difficult to verify the reliability of the neural network predictions.

For this paper, we are more interested in another class of data-driven approaches to tackle this problem, which use GP regression to allow for the analysis of the confidence of the prediction. The method of applying GPs to the CLF constraint was first introduced for closed-loop systems in [11]. Then, similar approaches have also been proposed for the construction of stability and safety constraints to be incorporated in min-norm controllers [12], [13], [14], [15].

However, all of these papers make an important assumption that might restrict their applicability, which is that the considered model uncertainty is unaffected by the control input. In contrast, for many controlled systems, uncertain input effects[1] are prevalent, e.g., in a mechanical system, uncertainty in the inertia matrix directly induces uncertain input effects. In the work presented in [16], a similar problem is addressed for the case of Control Barrier Function-based safety constraints [17] by the use of a Matrix-Variate GP regression. However, it does not provide a regression confidence analysis and results in an optimization problem that is not always convex. Finally, all the aforementioned GP-based approaches apply GP regression directly to the dynamics vector fields, which scale poorly with the system dimension.

In this paper, we develop solutions to overcome the presented limitations of the previous GP-based methods. First, we provide a formal way to deal with input-dependent model uncertainty of control-affine systems by proposing a specific GP kernel structure suitable for this problem. Since we apply GP regression to a scalar uncertainty term in the CLF constraint directly, compared to learning the uncertainty terms in the dynamics, we can reduce the computation of the regression significantly while still capturing many realistic forms of uncertainty. A similar kernel structure was used in [18] to learn the uncertainty terms in the autonomous and control vector fields separately for a single-input system. Here, we generalize the kernel to an arbitrary input dimension and derive expressions for the posterior GP of a combined input-dependent uncertainty term whose mean and variance are linear and quadratic in the input, respectively. By doing so, we can formulate a Second-Order Cone Program (SOCP) which incorporates a chance constraint that takes into account the confidence of the GP model and provides the exponential stabilizability of the system. We call it *Gaussian Process-based Control Lyapunov Function Second-*

*Indicates equal contribution.

Fernando Castañeda, Jason J. Choi, Bike Zhang, Claire J. Tomlin and Koushil Sreenath are with the University of California, Berkeley, CA, 94720, USA, {fcastaneda, jason.choi, bikezhang, tomlin, koushils}@berkeley.edu

This work was partially supported through National Science Foundation Grant CMMI-1931853 and DARPA Assured Autonomy program, grant FA8750-18-C-0101. The work of Fernando Castañeda received the support of a fellowship from Fundación Rafael del Pino, Spain. The work of Jason Choi received the support of a fellowship from Kwanjeong Educational Foundation, Korea.

[1]Uncertainty in the control vector field $g(x)$ in (1).

Order Cone Program (GP-CLF-SOCP). Formulation of the SOCP is crucial in that it can be solved quickly enough for real-time applications due to its convexity. Finally, since the inference time of GP regression is directly determined by the size of the training data, we maximize data efficiency by the use of an algorithm that iteratively collects data and improves the GP regression model in an episodic learning fashion.

The rest of the paper is organized as follows. In Section II, we give a brief overview of CLF-based controllers and show the effects that model uncertainty has on them. In Section III, we explain the basic concepts of GP regression. In Section IV, we present the compound kernel structure that allows us to regress the control-affine uncertainty. In Section V, we present the SOCP formulation of our uncertainty-aware optimization problem. In Section VI, we propose an efficient data-collection procedure. In Section VII we validate the proposed method for two different systems. Finally, in Section VIII, we provide concluding remarks.

II. Problem Statement

Throughout this paper we will consider nonlinear control-affine systems of the form

$$\dot{x} = f(x) + g(x)u, \tag{1}$$

where $x \in \mathbb{R}^n$ is the state of the system and $u \in \mathbb{R}^m$ is the control input. The vector fields $f\colon \mathbb{R}^n \to \mathbb{R}^n$ and $g\colon \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are assumed to be locally Lipschitz continuous and $f(0) = 0$.

The main objective of this paper is the construction of a locally stabilizing controller for such a system even when its dynamics are uncertain. A system is called *stabilizable* when it is asymptotically controllable to the origin with a feedback control law $u\colon \mathbb{R}^n \to \mathbb{R}^m$ that is continuous except possibly at the origin.

A. Control Lyapunov Functions

Definition 1. Let $V\colon \mathbb{R}^n \to \mathbb{R}_+$ be a positive definite, continuously differentiable and radially unbounded function. We say that V is a *Control Lyapunov Function* (CLF) for system (1) if for each $x \in \mathbb{R}^n \setminus \{0\}$

$$\inf_{u\in\mathbb{R}^m} \underbrace{L_fV(x) + L_gV(x)u}_{=\dot{V}(x,u)} < 0, \tag{2}$$

where the functions $L_fV(x) := \nabla V(x)\cdot f(x)$ and $L_gV(x) := \nabla V(x)\cdot g(x)$ are known as Lie derivatives.

If such a CLF exists, the system is known to be globally stabilizable [1]. Then, it is desirable to find a locally Lipschitz continuous feedback control law $u\colon \mathbb{R}^n \to \mathbb{R}^m$ such that the condition $L_fV(x) + L_gV(x)u(x) < 0$ holds for any $x \in \mathbb{R}^n \setminus \{0\}$. A simple way of synthesizing such a control law is by enforcing (2) as a constraint in a min-norm optimization problem. If u is unconstrained, this min-norm stabilizing controller can be expressed in closed-form [2].

However, many real-world systems require the addition of input constraints due to actuator limitations, i.e., $u \in U \subset \mathbb{R}^m$, in which case condition (2) becomes $\inf_{u\in U} L_fV(x) + L_gV(x)u < 0$, which might not be satisfied at every $x \in \mathbb{R}^n \setminus \{0\}$ even if V is a valid CLF for system (1). This fact motivates the following lemma.

Lemma 1. Let $V\colon \mathbb{R}^n \to \mathbb{R}_+$ be a CLF for system (1) and let $U \subset \mathbb{R}^m$ be the compact set of admissible control inputs. For each $c \in \mathbb{R}_+$ let Ω_c be the sublevel set of V such that $\Omega_c := \{x \in \mathbb{R}^n\colon V(x) \le c\}$. If there exists a $c_i > 0$ such that

$$\inf_{u\in U} L_fV(x) + L_gV(x)u < 0 \tag{3}$$

is satisfied $\forall x \in \Omega_{c_i} \setminus \{0\}$, then the system is locally stabilizable and Ω_{c_i} is a control invariant subset of the Region of Attraction (RoA) of the origin.

Proof. See [19, Proposition 2.2] for the proof of stabilizability. The control invariance proof is straightforward since for any x at the boundary of Ω_c we can always find a $u \in U$ such that $\dot{V}(x,u) < 0$ from condition (3). □

We can now take c_{max} as the maximum value of $c_i \in \mathbb{R}_+$ such that (3) holds for any $x \in \Omega_{c_i} \setminus \{0\}$. Then, $\Omega_{c_{max}}$ is the largest sublevel set of V contained in the RoA.

We can also consider a stronger notion of stabilizability by imposing exponential convergence to the origin. It is well-known that if there exists a compact subset $D \subseteq \Omega_{c_{max}}$ such that $\forall x \in D$ the following holds for some constant $\lambda > 0$,

$$\inf_{u\in U} L_fV(x) + L_gV(x)u + \lambda V(x) \le 0, \tag{4}$$

then the state of the system can be driven exponentially fast to the origin from any initial state $x_0 \in \Omega_{c_{exp}} \subseteq D$ [20]. If such $c_{exp} > 0$ exists, we will say that V is a *locally exponentially stabilizing* CLF. The condition (4) can be incorporated as a constraint into a min-norm optimization problem:

CLF-QP:

$$u^*(x) = \quad \arg\min_{u\in U} \quad u^Tu \tag{5a}$$

$$\text{s.t.} \quad L_fV(x) + L_gV(x)u + \lambda V(x) \le 0. \tag{5b}$$

In this paper, we assume that the input constraints are linear, which makes problem (5) a quadratic program (QP). We will refer to constraint (5b) as the *exponential CLF constraint*. This optimization problem defines a feedback control law $u^*\colon \mathbb{R}^n \to \mathbb{R}^m$ selecting the min-norm input such that the system state converges to the origin exponentially quickly. Note that, in practice, constraint (5b) is typically relaxed by adding a slack variable in order to guarantee the feasibility of the problem if condition (4) is not locally satisfied [3].

B. Effects of Model Uncertainty on CLF-based Controllers

The main problem concerned in this paper is how to reformulate the min-norm stabilizing controller defined in (5) in the presence of model uncertainty.

First, we provide some necessary settings and assumptions for our problem formulation. Let's assume that we have a *nominal model*

$$\dot{x} = \tilde{f}(x) + \tilde{g}(x)u, \tag{6}$$

where $\tilde{f} : \mathbb{R}^n \to \mathbb{R}^n$, $\tilde{g} : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are Lipschitz continuous vector fields and $\tilde{f}(0) = 0$. We assume that we have a locally exponentially stabilizing CLF V for the nominal model, and that the plant is also locally exponentially stabilizable with the same V. Note that the region of exponential stabilizablity around the origin can be sufficiently small. Also, the assumption can be relaxed to asymptotic stabilizability if the user is concerned with enforcing condition (3) instead of (4). In general, $\tilde{f}$ and $\tilde{g}$ would be different from the true plant vector fields f and g because the nominal model is imperfect. The assumption implies, however, that they share some similarity through the stabilizing property of the same function V. Finally, we also assume that we have access to measurements of state and control input at every sampling time Δt.

Our main objective is to construct the exponential CLF constraint (5b) for the true plant when we only know the model dynamics $\tilde{f}$ and $\tilde{g}$. Since $\dot{V}(x,u) = L_f V(x) + L_g V(x)u$ depends on the dynamics of the plant, the estimate based on the nominal model $\dot{\tilde{V}}(x,u) = L_{\tilde{f}}V(x) + L_{\tilde{g}}V(x)u$, will differ from its true value. We define $\Delta : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ as the difference between these:

$$\Delta(x,u) := \dot{V}(x,u) - \dot{\tilde{V}}(x,u). \tag{7}$$

Then, the exponential CLF constraint (5b) becomes

$$L_{\tilde{f}}V(x) + L_{\tilde{g}}V(x)u + \Delta(x,u) + \lambda V(x) \leq 0. \tag{8}$$

Therefore, verifying the exponential CLF constraint for the true plant amounts to a problem of learning the mismatch term $\Delta(x,u)$ correctly and then enforcing (8). We can learn this function from the past data by formulating a supervised learning problem. Specifically, we will use GP regression, a method that will be introduced in the next section.

Remark 1. In (7), if we express $\dot{V}$ and $\dot{\tilde{V}}$ with their respective Lie derivatives, we get

$$\Delta(x,u) = \underbrace{(L_f V(x) - L_{\tilde{f}}V(x))}_{=:\Delta_1(x)} + \underbrace{(L_g V(x) - L_{\tilde{g}}V(x))}_{=:\Delta_2(x)} u. \tag{9}$$

Note that we do not have access to $\Delta_1(x)$ and $\Delta_2(x)$ in this equation since we are unaware of f and g. It is tempting to learn each of these terms separately with supervised learning. However, we can only measure $\Delta(x,u)$, which makes this approach intractable. Nevertheless, we can exploit the fact that the mismatch term $\Delta(x,u)$ is control-affine.

III. Gaussian Process Regression

A Gaussian Process is a random process such that any finite selection of samples $\{h(x_k)\}_{k=1}^n$ has a joint Gaussian distribution. A GP is fully determined by its mean function $q : \mathcal{X} \to \mathbb{R}$ and covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, i.e.,

$$h(x) \sim \mathcal{GP}(q(x), k(x,x')), \tag{10}$$

where $\mathcal{X}$ is the input domain, a connected subset of $\mathbb{R}^n$, $h(x)$ is the output function sampled from the GP, and x, x' denote input variables in $\mathcal{X}$. Any positive definite kernel function[2] can be a valid covariance function [21]. Such a kernel $k(x,x')$ can be used to generate a set of functions that satisfy a specific property, namely a "reproducing" property. An inner product between such a function h and the kernel $k(\cdot,x)$ should reproduce h, i.e., $\langle h(\cdot), k(\cdot,x)\rangle = h(x),\ \forall x \in \mathcal{X}$. Such a set is called a Reproducing Kernel Hilbert Space (RKHS, [21]), a specific class of function space, and is denoted as $\mathcal{H}_k(\mathcal{X})$. The RKHS norm $\|h\|_k := \sqrt{\langle h, h\rangle}$, which will be used in Lemma 2, is a measure of the smoothness of h with respect to the kernel function[3]. Note that an appropriate inner product in the above expressions would be determined by the specific choice of the associated reproducing kernel k.

GPs encode prior distributions over functions and given new query points, a posterior distribution can be derived from the joint Gaussian distribution between the prior data and the query points. This gives rise to their typical application in the machine learning literature: GP regression. For the remainder of the paper, we use $q(x) \equiv 0$ as the mean function of the prior GP. Given N input-output data pairs $\{(x_j, z_j)\}_{j=1}^N$, the regressor is provided by the posterior GP distribution conditioned on the data. Here, the output data is assumed to be measurements of $h(x_j)$, where h is the target function for regression, with additive white noise, i.e., $z_j = h(x_j) + \varepsilon_j$, where $\varepsilon_j \sim \mathcal{N}(0, \sigma_n^2)$. Then, the mean and the variance of the posterior $h(x_*)$ at a query point x_*, are given as

$$\mu_* = \mathbf{z}^T (K + \sigma_n^2 I)^{-1} K_*^T, \tag{11}$$

$$\sigma_*^2 = k(x_*, x_*) - K_* (K + \sigma_n^2 I)^{-1} K_*^T, \tag{12}$$

which are derived from the distribution of $h(x_*)$ conditioned on $\{z_j\}_{j=1}^N$ [22]. $K \in \mathbb{R}^{N\times N}$ is the Gram matrix whose (i^{th}, j^{th}) element is defined as $k(x_i, x_j)$ for $i, j = 1, \cdots, N$, and $K_* = [k(x_*, x_1),\ \cdots\ , k(x_*, x_N)] \in \mathbb{R}^{1\times N}$. $\mathbf{z} \in \mathbb{R}^N$ is the vector containing the output measurements z_j. Note that there exist various choices of kernel functions and many of them depend on some hyperparameters which determine the kernel's characteristics. Depending on the choice of kernel and hyperparameters, the result of the regression varies, and the problem of choosing the best kernel and its hyperparameters is known as the "training" process of the GP regression [22]. In this work we use marginal likelihood maximization, which is one of the most common training methods.

After training, one would like to study how close the GP model approximates the target function. In order to do this, we use the Upper Confidence Bound (UCB) analysis [23], specifically, the following lemma.

Lemma 2. [23, Thm. 6] Assume that the noise sequence $\{\varepsilon_j\}_{j=1}^\infty$ is zero-mean and uniformly bounded by σ_n. Let

[2] k is a positive definite kernel if its associated kernel matrix $K(x_1, x_2)$, whose (i^{th}, j^{th}) element is defined as $k(x_i, x_j)$, is positive semi-definite for any distinct points $x_1, x_2 \in \mathcal{X}$.

[3] $\|h(x) - h(x')\|_2 \leq \|h\|_k \, \|k(x,\cdot) - k(x',\cdot)\|_k \ \forall x, x' \in \mathcal{X}$

the target function $h : \mathcal{X} \to \mathbb{R}$ be a member of $\mathcal{H}_k(\mathcal{X})$ associated with a bounded kernel k, with its RKHS norm bounded by B. Then, with probability of at least $1-\delta$, the following holds for all $x \in \mathcal{X}$ and $N \geq 1$:

$$|\mu_* - h(x_*)| \leq \left(2B^2 + 300\gamma_{N+1}\ln^3((N+1)/\delta)\right)^{0.5}\sigma_*,$$

where γ_{N+1} is the maximum information gain after getting $N+1$ data points, and μ_*, σ_*^2 are the mean and variance of the posterior GP given by (11) and (12).

Proof. See [23, Thm. 6]. □

In this lemma, the assumption about the boundedness of $\|h\|_k$ implicitly requires a "low complexity" of the target function [24]. B is usually unknown a priori, but a trial-and-error approach to find its value suffices in practice [23]. γ_{N+1} quantifies the reduction of uncertainty about h in terms of entropy. It has a sublinear dependency on N for many commonly used kernels and it can be efficiently approximated up to a constant [23].

IV. GP Regression for Affine Target Functions

In this section, we use GP regression to learn the mismatch term $\Delta(x,u)$ (7) from data. From (9), we know that $\Delta(x,u)$ is affine in u. If we use an arbitrary kernel, we cannot exploit this information in the GP regression. Therefore, our first objective is to construct an appropriate kernel that captures the control-affine structure of Δ in the regression. In order to do this, we introduce the general formulation of this problem in this section. Consider p functions, $h_i : \mathcal{X} \to \mathbb{R}$ for $i = 1, \cdots, p$, and define

$$h_c(x,y) \coloneqq [h_1(x)\ h_2(x)\ \cdots\ h_p(x)] \cdot y, \tag{13}$$

where $y \in \mathcal{Y} \subset \mathbb{R}^p$. Our objective is to estimate the function $h_c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ which is affine in y by using GP regression, given its measurements $z_j = h_c(x_j, y_j) + \varepsilon_j$ for $j = 1, \cdots, N$.

The underlying structure of $h_c(x,y)$ tells us that it contains information about p random functions $\{h_i(x)\}_{i=1}^p$ condensed to a single scalar value by a dot product with y. Therefore, it is natural to consider p underlying kernels and their composition. For $i = 1, \cdots, p$, consider covariance functions $k_i : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

Definition 2. *Affine Dot Product Compound Kernel:* Define k_c given by

$$k_c\left(\begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} x' \\ y' \end{bmatrix}\right) \coloneqq y^T Diag([k_1(x,x'), \cdots, k_p(x,x')])y', \tag{14}$$

as the *Affine Dot Product* (ADP) compound kernel of p individual kernels $k_1(x,x')$, $\cdots$, $k_p(x,x')$.

Note that for a fixed (x,x'), the ADP compound kernel resembles the well-known dot product kernel, defined as $k(y,y') = y^T y'$ [22].

Lemma 3. If $k_1, \cdots, k_p$ are positive definite kernels, the ADP compound kernel k_c is also positive definite. Furthermore, if $k_1, \cdots, k_p$ are bounded kernels, k_c is also bounded.

Proof. Consider the Gram matrix of k_c, $K_c \in \mathbb{R}^{N\times N}$ for $\{(x_j, y_j)\}_{j=1}^N$. Let K_i be the Gram matrix of k_i for $\{x_j\}_{j=1}^N$. Define $Y \coloneqq [y_1\ y_2\ \cdots\ y_N] \in \mathbb{R}^{p\times N}$, and let $\mathbf{y_i}^T$ be the i-th row of Y. Then,

$$K_c = \sum_{i=1}^{p} \left(\mathbf{y_i}\ \mathbf{y_i}^T\right) \circ K_i,$$

where $\circ$ indicates the Hadamard product [25]. By the Schur Product Theorem [25], if the k_i are positive definite kernels, then since each $\mathbf{y_i}\mathbf{y_i}^T$ and K_i are positive semidefinite, K_c is a positive semidefinite matrix. Therefore, k_c is a positive definite kernel by definition. Also, if the k_i are bounded kernels, each K_i is bounded so K_c is also bounded. Therefore, k_c is a bounded kernel. □

By Lemma 3, since k_c is positive definite, it is a valid covariance function. Consider a set of functions $\mathcal{H}_{k_c}(\mathcal{X}\times\mathcal{Y}) \coloneqq \{h_c : \mathcal{X}\times\mathcal{Y} \to \mathbb{R} \mid \exists h_i \in \mathcal{H}_{k_i}$ for $i = 1, \cdots, p$, s.t. $h_c(x,y) = [h_1(x), \cdots, h_p(x)] \cdot y\}$ where each $\mathcal{H}_{k_i}$ is the RKHS whose reproducing kernel is k_i. Then, the following holds:

Theorem 1. $\mathcal{H}_{k_c}(\mathcal{X}\times\mathcal{Y})$ is an RKHS whose reproducing kernel is k_c in Definition 2.

Proof. Define the inner product of $\mathcal{H}_{k_c}$ to be

$$\langle h_c, h_c' \rangle_c \coloneqq \sum_{i=1}^{p} \langle h_i, h_i' \rangle_i,$$

for $\forall h_c, h_c' \in \mathcal{H}_{k_c}$ where $\{h_i\}_{i=1}^p$ and $\{h_i'\}_{i=1}^p$ are sets of functions whose i-th elements are from $\mathcal{H}_{k_i}$ that satisfy $h_c(x,y) = [h_1(x), \cdots, h_p(x)] \cdot y$ and $h_c'(x,y) = [h_1'(x), \cdots, h_p'(x)] \cdot y$, respectively. Such sets of functions should exist by definition of $\mathcal{H}_{k_c}$. $\langle h_i, h_i' \rangle_i$ is the inner product of $\mathcal{H}_{k_i}$. It is trivial that this definition satisfies the axioms of the inner product. Then,

$$\begin{aligned}\left\langle h_c(\cdot,\blacksquare), k_c\left(\begin{bmatrix}\cdot\\ \blacksquare\end{bmatrix}, \begin{bmatrix} x \\ y \end{bmatrix}\right)\right\rangle_c &= \sum_{i=1}^{p} y_i \langle h_i(\cdot), k_i(\cdot, x)\rangle_i \\ &= \sum_{i=1}^{p} y_i h_i(x) = h_c(x,y).\end{aligned}$$

The first equality holds because of Definition 2 and the definition of $\langle\cdot,\cdot\rangle_c$. The second equality holds because of the reproducing property of each $k_i(\cdot,\cdot)$. □

Theorem 1 allows us to apply the UCB analysis from Section III to $h_c(x,y)$ with some additional conditions which will be specified in Section V. Regression for $h_c(x,y)$ in Lemma 2 (i.e. μ_*, σ_*) now can be treated in the same way as any other kind of general GP regression, but with a specific choice of covariance function given by (14).

One caveat of this regression is that depending on the distribution of the inputs y_j in the data, this problem can be underdetermined. For instance, when every y_j is a constant vector, there are infinitely many choices of valid $h_i(x)$ that give the same estimation error. Nevertheless, under our GP regression structure, this evidence of underdetermination is implicitly captured by larger values of the variance of the

posterior. In practice, it is preferable to avoid such underdetermination since we want to reduce the uncertainty of the GP posterior. Therefore, we need to carefully collect the training data to make sure we capture rich enough information about the target function. In Section VI we propose a method for this purpose. In the system identification literature, this is related to the property of persistency of excitation [26].

Finally, the main benefit of exploiting the affine structure in the kernel is revealed in the expressions for the posterior distribution's mean and variance. This is the main difference in how we use the ADP kernel compared to [18], where a similar kernel is proposed for a special case $p = 2$. Let $X \in \mathbb{R}^{n\times N}$, $Y \in \mathbb{R}^{p\times N}$ be matrices whose column vectors are the inputs x_j and y_j of the collected data, respectively, and let $\mathbf{z} \in \mathbb{R}^N$ be the vector containing the output measurements z_j. Then, plugging them and the ADP compound kernel into (11) and (12) gives the following expressions for the mean and variance of the posterior at a query point (x_*, y_*):

$$\mu_* = \underbrace{\mathbf{z}^T(K_c + \sigma_n^2 I)^{-1}K_{*Y}^T}_{=:b_*^T} y_*, \tag{15}$$

$$\sigma_*^2 = y_*^T \underbrace{\left(Diag\left(\begin{bmatrix} k_1(x_*,x_*) \\ \vdots \\ k_p(x_*,x_*) \end{bmatrix}\right) - K_{*Y}(K_c + \sigma_n^2 I)^{-1}K_{*Y}^T \right)}_{=:C_*} y_*. \tag{16}$$

Here, $K_c \in \mathbb{R}^{N\times N}$ is the Gram matrix of k_c for the training data inputs (X, Y), and $K_{*Y} \in \mathbb{R}^{p\times N}$ is given by

$$K_{*Y} = \begin{bmatrix} K_{1*} \\ K_{2*} \\ \vdots \\ K_{p*} \end{bmatrix} \circ Y,\ K_{i*} = [k_i(x_*, x_1),\ \cdots\ , k_i(x_*, x_N)].$$

Readers can observe that (15) and (16) are affine and quadratic in y_*, respectively. These structures are critical when formulating the uncertainty-aware CLF chance constraint as a second-order cone constraint in the next section.

V. Uncertainty-Aware Min-norm Stabilizing Controller

A. Probabilistic Bounds on the CLF Derivative

We have already presented all the necessary tools to verify the probabilistic bounds on the mismatch term $\Delta(x, u)$ in (9). Indeed, learning Δ corresponds to the GP regression problem defined by (15), (16), in which the target function h_c is Δ, x is the state, $y = [1, u^T]^T$, $p = m + 1$, h_1 is Δ_1, and h_{i+1} is Δ_2's i-th element for $i = 1, \cdots, m$.

Assumption 1. Consider bounded reproducing kernels k_i for $i = 1, \cdots, m+1$. We assume that Δ_1 is a member of $\mathcal{H}_{k_1}$ and each i-th element of Δ_2 is a member of $\mathcal{H}_{k_{i+1}}$ for $i = 1, \cdots, m$, respectively. We assume that their RKHS norms are bounded.

Lemma 4. Under Assumption 1 and with a compact set of admissible control inputs U, Δ is a member of $\mathcal{H}_{k_c}$, the RKHS created by the ADP compound kernel of k_i for $i = 1, \cdots, m+1$. Moreover, its RKHS norm is bounded, namely $\|\Delta\|_{k_c} \leq B$.

Proof. The proof follows from Thm. 1 and the definition of the inner product for $\mathcal{H}_{k_c}$ in the proof of Thm. 1. □

Assumption 2. We have access to measurements $z_i = \dot{V}(x_i, u_i) - (L_{\tilde{f}}V(x) + L_{\tilde{g}}V(x)u_i) + \varepsilon_i$, and the noise term ε_i is zero-mean and uniformly bounded by σ_n.

With Assumptions 1, 2 and Lemma 4, we can now apply Lemma 2 to our regression problem.

Theorem 2. Let Assumptions 1 and 2 hold. Let $\beta := \left(2B^2 + 300\gamma_{N+1}\ln^3((N+1)/\delta)\right)^{0.5}$, with N the number of data points, and γ_{N+1} as defined in Lemma 2. Let μ_* and σ_*^2 be the mean and variance of the posterior for Δ using the ADP compound kernel, at a query point (x_*, u_*) as obtained from (15) and (16). Then, with a probability of at least $1 - \delta$ the following holds:

$$|\mu_* - \Delta(x_*, u_*)| \leq \beta\sigma_*. \tag{17}$$

Proof. Proof follows from Lemmas 2 and 4. □

The error in the estimation of the mismatch term Δ is now bounded for some confidence level. From (17) we can easily derive the bounds on the true derivative of the CLF for a probability of at least $1 - \delta$:

$$\tilde{\dot{V}}(x_*, u_*) + \mu_* - \beta\sigma_* \leq \dot{V}(x_*, u_*) \leq \tilde{\dot{V}}(x_*, u_*) + \mu_* + \beta\sigma_*. \tag{18}$$

B. GP-Based CLF Second-Order Cone Program

Taking the upper bound of (18), we can enforce the exponential CLF constraint of (5b) with a probability of at least $1 - \delta$, and incorporate the resulting chance constraint into a min-norm optimization problem that defines a feedback control law $u^*\colon \mathbb{R}^n \to \mathbb{R}^m$ pointwise:

GP-CLF-SOCP:

$$\begin{aligned} u^*(x) = \quad & \underset{u\in U,\ d\in\mathbb{R}}{\arg\min} \quad u^T u + p\ d^2 \\ \text{s.t.} \quad & \tilde{\dot{V}}(x, u) + \mu_*(x, u) + \beta\sigma_*(x, u) + \lambda V(x) \leq d. \end{aligned} \tag{19}$$

With a slight abuse of notation, $\mu_*(x, u)$ and $\sigma_*(x, u)$ are the mean and standard deviation of the GP posterior at (x, u), obtained from (15) and (16).

Remark 2. The stability constraint is relaxed in order to guarantee the feasibility of the problem. If the initial state x_0 is outside the CLF maximum sublevel set for exponential stability $\Omega_{c_{exp}}$, we cannot guarantee exponential convergence and neither can the controller which uses the true plant dynamics. However, even for this case, we still do guarantee that the approximation error of the CLF derivative is bounded as given by (18) with probability $1 - \delta$.

Note that this optimization problem does not require knowledge about the true plant dynamics. The fact that μ_* and σ_*^2 are affine and quadratic in u, respectively, is crucial for the following main result of the paper:

Theorem 3. Using the proposed ADP compound kernel from Definition 2, the uncertainty-aware optimization problem (19) is convex, meaning that its global minimum can be reliably recovered. Specifically, it is a Second-Order Cone Program (SOCP).

Proof. Let's first transform the quadratic objective function into a second-order cone constraint and a linear objective. Let the objective function be $J(u,d) \coloneqq u^T u + p\ d^2$. Note that by taking $\varphi = [u^T, d]^T$ we can express the objective as $J(\varphi) = \varphi^T Q \varphi$. Next, by setting $z \coloneqq L\varphi$, where L is the matrix square-root of Q, we can rewrite $J(z) = \|z\|_2^2$. Note that minimizing J gives the same result as minimizing $J'(z) \coloneqq \|z\|_2$. Now we can move the objective function J' into a second-order cone constraint by setting $\|z\|_2 \leq t$ and minimizing the new linear objective function $J''(t) \coloneqq t$.

The next step is to prove that the CLF chance constraint is a second-order cone constraint. Note that $\dot{\tilde{V}}(x,u) = L_{\tilde{f}}V(x) + L_{\tilde{g}}V(x)u$ and $\mu_*(x,u) = b_*^T[1,u^T]^T$ are both control-affine. Note that $\sigma_*(x,u) = \sqrt{[1,u^T]C_*(x)[1,u^T]^T}$ can be rewritten as $\sigma_*(x,u) = \|M(x)u + n(x)\|_2$, although we omit the expressions of M and n for conciseness. Therefore, the CLF chance constraint is a second-order cone constraint, and the resulting optimization problem is an SOCP with two second-order cone constraints corresponding to the original objective function and the CLF chance constraint. SOCPs are inherently convex. □

VI. Data Collection

In this section, we introduce an algorithm that efficiently collects measurements of Δ for the GP regression. This data should contain rich enough information about Δ, especially about its dependency on u as discussed in Section IV, and since our goal is to obtain a locally stabilizing controller, it is preferable to exclude the data from outside the RoA for efficiency. To this end, we propose an algorithm that iteratively collects new data and trains a new GP model in an episodic learning fashion. The algorithm uses the level sets of the CLF as "guides" for expanding the training region by exploiting Lemma 1. In addition, we use the idea of greedy search in the Bayesian Optimization literature [23] to actively explore the most uncertain area of the training region. Our algorithm is based on the active learning algorithm of [27], although while [27] focuses on guaranteeing safety online, our objective is to maximize the efficiency of the offline data collection.

A. Discrete-Time Measurements

First consider how to obtain inputs (x_j, u_j) and labels (z_j) —measurements of $\Delta(x_j,u_j)$— of the training data. Let $x(t)$ and $u(t)$ be the state and control input measurements at time t and $x(t+\Delta t)$ be the state measurement at the next timestep. We can use these values to create input-label pairs with $\mathcal{O}(\Delta t^2)$ approximation error:

$$x_j = \frac{x(t+\Delta t) + x(t)}{2}, \quad u_j = u(t),$$
$$z_j = \frac{V(x(t+\Delta t)) - V(x(t))}{\Delta t} - \dot{\tilde{V}}(x_j, u_j).$$

Note that u_j is the control input during the interval $[t, t+\Delta t)$, and z_j is the difference between the value of $\dot{V}(x_j,u_j)$ obtained from numerical differentiation and the nominal model-based estimate.

B. Estimation of the Region of Attraction

Next, we introduce a new certificate with the learned uncertainty for a conservative estimation of the RoA. Notice that the condition for inclusion in the RoA provided by Lemma 1, is only valid when there is no model-plant mismatch. Thus, we have to incorporate the learned uncertainty terms from Section IV as we do when we formulate the GP-CLF-SOCP in Section V.

Theorem 4. Taking the GP posterior distribution from the training data $\{(x_j,u_j,z_j)\}_{j=1}^N$, and β from (17), if there exists a $c > 0$ such that for all $x \in \Omega_c$ it holds that

$$\inf_{u \in U} \ \dot{\tilde{V}}(x,u) + \mu_*(x,u) + \beta\sigma_*(x,u) < 0, \qquad (20)$$

then Ω_c is in the RoA with probability at least $(1-\delta)$.

Proof. Proof follows from Lemma 1 and Theorem 2. □

Notice that this certificate is "conservative" in the sense that it takes the worst-case bound of the effect of the uncertainty term, based on the collected data. Therefore, if we collect more data and improve our GP model to have less uncertainty, then the conservatism will reduce and we will be able to obtain a bigger subset of the RoA. This is the central principle of the algorithm.

C. Algorithm Overview

Finally, we give an overview of the proposed algorithm.

1) Initial GP Model: We start by considering a level set Ω_{c_0} which is small-enough to be a subset of the RoA (Fig 1.a). Such $c_0 > 0$ always exists due to our assumption that V is a locally valid CLF. We collect an initial batch of training data (D_0) from a set of trajectories whose initial states are randomly sampled from Ω_{c_0}, and train an initial GP regression model. Here, we use the nominal model-based CLF-QP from (5) as our stabilizing controller.

2) Episodic Learning: The main loop of our algorithm consists of a series of episodes, and each i-th episode is mainly composed of three steps. **1)** In the first step (Fig. 1.b), we obtain a set of N_e points from $(\Omega_{(c_{i-1}+\Delta c_i)} \backslash \Omega_{c_{i-1}}) \times \mathcal{U}$ at which the variance of the posterior of the current GP model is maximal. Δc_i is the parameter that determines the size of the new exploration region. **2)** Next (Fig 1.c), we run short rollouts by taking each point from Step 1 as our initial state and initial control input. During the rollouts, we also evaluate the stabilizability condition (20) at each timestep. Note that such evaluation is a feasibility problem which is also an SOCP since (20) is a second-order cone constraint. After the rollouts, we expand the level of V (we determine c_i) up to a point for which (20) becomes infeasible. **3)** Finally (Fig 1.d), we add the data obtained from the trajectories within Ω_{c_i} to our data set, and train the next GP regression model.

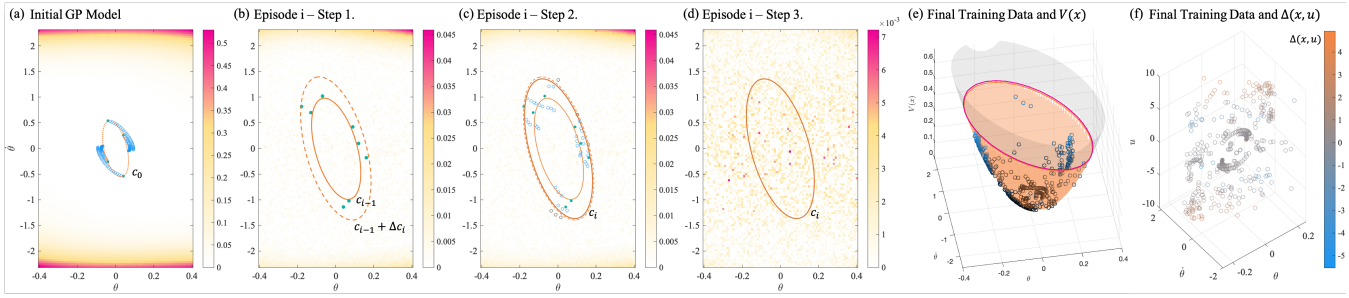

Fig. 1. **(a–d): Visualization of the episodic learning data collection algorithm running on the inverted pendulum example:** Color map represents the maximum variance of the posterior GP, $\max_{u\in U} \Delta_*(x,u)$. Orange curves: level curves of the CLF. Green points: initial states for the rollouts, Blue points: trajectory points added to the training data. Grey points: trajectory points excluded from the training data since they are outside Ω_{c_i}. **(a) Initial GP Model:** Trajectories sampled from the initial level set Ω_{c_0} by running the CLF-QP are collected to create an initial GP model. **(b) Episode i-Step 1:** N_e initial states and initial control inputs in $(\Omega_{(c_{i-1}+\Delta c_i)} \setminus \Omega_{c_{i-1}}) \times \mathcal{U}$ are determined where σ_* are maximal. **(c) Episode i-Step 2:** Simulations are run from such initial points and the resulting trajectories are saved. At the same time, c_i is determined by evaluating (20) for the sampled trajectories. **(d) Episode i-Step 3:** Finally, the i-th GP model is updated. Note the reduction in the variance. (Total episodes = 7, $i = 3$ for (b), (c), (d).) **(e, f): Distribution of the final training data** plotted in the x–$V(x)$ space (blue points) and plotted in x–u space, respectively. (e) Level curve in color magenta is the $\Omega_{c_{max}}$ (maximum level set contained in the RoA) for the true plant. The value of CLF is plotted in grey and the orange region is the region verified as RoA through the data collection algorithm. (f) The color indicates the value of z_i, the measurement of $\Delta(x_i, u_i)$. The number of data points is 425.

Remark 3. In Step 2 of an episode, we check condition (20) only for finite sampled states in $\Omega_{c_i} \setminus \Omega_{c_{i-1}}$, whereas Theorem 4 requires (20) to be satisfied at every state in Ω_{c_i}. Notice that brute-force verification for the whole region of Ω_{c_i} will scale poorly with state dimension. Even though we do not have the rigorous guarantee of Theorem 4 with this algorithm, the error in the estimated c_{max} does not affect the probabilistic guarantee of the resulting GP-CLF-SOCP controller. In practice, we observe that we can well approximate c_{max} such that $\Omega_{c_{max}}$ is contained in the true RoA (See Fig. 1(e)).

VII. Examples

A. Two-dimensional System: Inverted Pendulum

Consider a control-affine two-dimensional inverted pendulum as the one in [27], with parameters of the plant $m_{\text{plant}} = 2$kg, $l = 1$m and for the model, $m_{\text{model}} = 1$kg, $l = 1$m, which results in model uncertainty in both f and g in (1).

A CLF-QP controller (5) based on the nominal model is designed to stabilize the pendulum to the upright position. In order to illustrate the effects of model uncertainty, we compare it with the CLF-QP controller based on the true plant dynamics. The difference between the two controllers (Fig. 2) is due to the effects of model uncertainty. Specifically, in this case the model uncertainty makes the system converge more slowly.

Fig. 1 depicts the data collection algorithm and the resulting training data for the GP model. The results of deploying the GP-CLF-SOCP controller, with a confidence level of $1 - \delta = 0.95$, are presented in Fig. 2 in blue lines. Note that the results are very similar to those from the CLF-QP based on the true plant dynamics, which means that the GP-CLF-SOCP successfully captures the correct effects of model uncertainty. Also, the computation time of the GP-CLF-SOCP, including the GP inference time, is 9.1 ± 2.2ms (max: 25.7ms) on a laptop with a 10th-gen Intel Core i7 and 32GB RAM.

In order to benchmark the GP-CLF-SOCP, we compare its performance with the one obtained if we only learn the uncertainty in f, as done in previous works [13], [15]. For this, we design a GP-based Control Lyapunov Function Quadratic Program (GP-CLF-QP) that only learns the uncertainty Δ_1 in (9). The results of this controller are also shown in Fig. 2.

B. System with Multiple Control Inputs: Kinematic Bicycle

Next, in order to show that our method can be successfully applied to systems with higher state dimension and multiple control inputs, we apply it to track a reference trajectory using a kinematic bicycle model. The state is defined as $x = [p_x, p_y, v, \theta, \gamma]^T$ (p_x, p_y: position coordinates, v: speed, θ: heading angle, γ: tangent of the steering angle). The dynamics of the system are given as

$$\dot{x} = f(x) + g(x)u, \quad f(x) = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ -f_\mu \\ v\gamma \\ 0 \end{bmatrix}, \; g(x) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ b_v & 0 \\ 0 & 0 \\ 0 & b_\gamma \end{bmatrix}, \quad (21)$$

where $u \in \mathbb{R}^2$, and f_μ, b_v, b_γ are constants that emulate friction and skid effects. For the nominal model, we assume no such effects ($f_\mu = 0$, $b_v = b_\gamma = 1$) and for the plant, we use $f_\mu = 1$, $b_v = 1.5$, $b_\gamma = 0.75$. The objective is to stabilize to a constant-velocity trajectory along the x-axis; $v(t) = 5$, $p_y(t) = \theta(t) = \gamma(t) = 0$. The initial state is set as $x_0 = [0, 0.25, 2, 0.25, 0.25]^T$.

Fig. 3 shows the simulation results of the GP-CLF-SOCP and those of the CLF-QP based on the nominal model and the true plant. Here, we use a polynomial CLF [28], which is verified to be locally stabilizing for the nominal model. While the nominal model-based CLF-QP oscillates around the reference trajectory, the GP-CLF-SOCP successfully converges to the reference trajectory.

VIII. Conclusion

We have presented a method to design a stabilizing controller for control-affine systems with both state and input-dependent model uncertainty using GP regression. For this purpose, we have proposed the novel ADP compound kernel, which captures the control-affine nature of the problem.

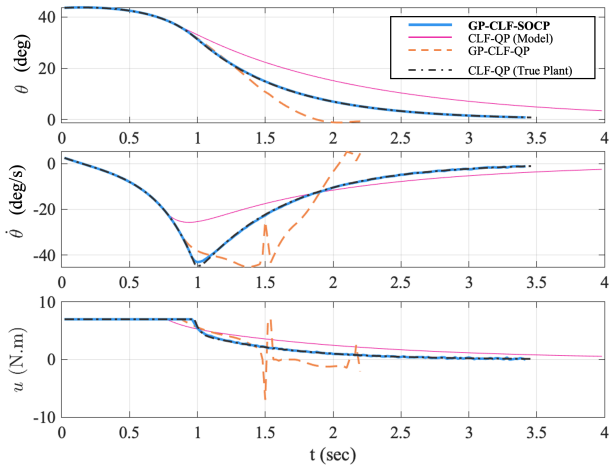

Fig. 2. Simulation results of applying the GP-CLF-SOCP to the inverted pendulum example, with a model-plant mismatch of m_{plant} =2kg, m_{model} =1kg, compared to the nominal-model-based CLF-QP, and to the GP-CLF-QP that does not consider the uncertainty affected by u. Results of the CLF-QP based on the true plant are also provided to show that the GP-CLF-SOCP learns the correct exponential CLF constraint.

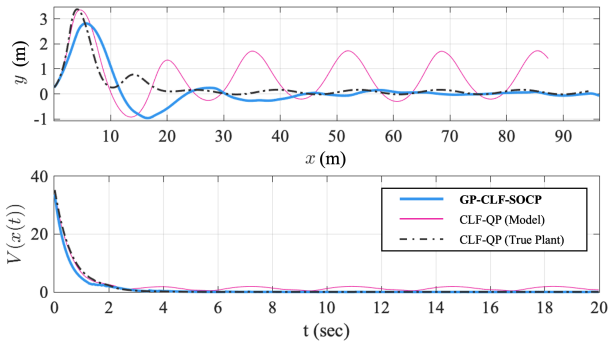

Fig. 3. Trajectories in $x-y$ plane (Top) and histories of $V(x(t))$ (Bottom) of the kinematic vehicle under artificial drift and friction to illustrate the applicability of the GP-CLF-SOCP to multi-input systems. Comparison between GP-CLF-SOCP, CLF-QP(Model), and CLF-QP(Plant). The sampling time is set as 20ms, and the comptutation time of the GP-CLF-SOCP per timestep is 10.3 ± 1.9ms (max: 20.4ms). Number of training data points for GP-CLF-SOCP: 961.

This permits the formulation of the so-called GP-CLF-SOCP, which is solved online to obtain an exponentially stabilizing controller with probabilistic guarantees. After testing it on the numerical simulation of two different systems, we obtain a clear improvement with respect to the nominal model-based CLF-QP and we are able to closely match the performance of the true plant-based controller.

REFERENCES

[1] Z. Artstein, "Stabilization with relaxed controls," *Nonlinear Analysis: Theory, Methods and Applications*, vol. 7, pp. 1163 – 1173, 1983.

[2] E. D. Sontag, "A 'universal' construction of artstein's theorem on nonlinear stabilization," *Systems and Control Letters*, vol. 13, no. 2, pp. 117 – 123, 1989.

[3] K. Galloway, K. Sreenath, A. D. Ames, and J. W. Grizzle, "Torque saturation in bipedal robotic walking through control lyapunov function-based quadratic programs," *IEEE Access*, vol. 3, pp. 323–332, 2015.

[4] Q. Nguyen and K. Sreenath, "Optimal robust control for bipedal robots through control lyapunov function based quadratic programs." in *Robotics: Science and Systems*. Rome, Italy, 2015.

[5] J. Reher, C. Kann, and A. D. Ames, "An inverse dynamics approach to control lyapunov functions," in *American Control Conference*, 2020.

[6] A. D. Ames and M. Powell, "Towards the unification of locomotion and manipulation through control lyapunov functions and quadratic programs," in *Control of Cyber-Physical Systems*. Springer, 2013, pp. 219–240.

[7] Q. Nguyen and K. Sreenath, "L1 adaptive control for bipedal robots with control lyapunov function based quadratic programs," in *American Control Conference*, Chicago, IL, July 2015, pp. 862–867.

[8] A. J. Taylor, V. D. Dorobantu, H. M. Le, Y. Yue, and A. D. Ames, "Episodic learning with control lyapunov functions for uncertain robotic systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 6878–6884.

[9] J. Choi, F. Castañeda, C. Tomlin, and K. Sreenath, "Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions," in *Robotics: Science and Systems*, Corvalis, OR, July 2020.

[10] T. Westenbroek, F. Castañeda, A. Agrawal, S. S. Sastry, and K. Sreenath, "Learning min-norm stabilizing control laws for systems with unknown dynamics," in *IEEE Conference on Decision and Control*, 2020, pp. 737–744.

[11] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes," in *IEEE Conference on Decision and Control*, 2016, pp. 4661–4666.

[12] J. Umlauft, L. Pöhler, and S. Hirche, "An uncertainty-based control lyapunov approach for control-affine systems modeled by gaussian process," *IEEE Control Systems Letters*, vol. 2, pp. 483–488, 2018.

[13] D. D. Fan, J. Nguyen, R. Thakker, N. Alatur, A. a. Agha-mohammadi, and E. A. Theodorou, "Bayesian learning-based adaptive control for safety critical systems," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 4093–4099.

[14] R. Cheng, M. J. Khojasteh, A. D. Ames, and J. W. Burdick, "Safe multi-agent interaction through robust control barrier functions with learned uncertainties," *arXiv preprint arXiv:2004.05273*, 2020.

[15] L. Zheng, R. Yang, J. Pan, H. Cheng, and H. Hu, "Learning-based safety-stability-driven control for safety-critical systems under model uncertainties," in *International Conference on Wireless Communications and Signal Processing*, 2020, pp. 1112–1118.

[16] M. J. Khojasteh, V. Dhiman, M. Franceschetti, and N. Atanasov, "Probabilistic safety constraints for learned high relative degree system dynamics," in *Learning for Dynamics and Control*, 2020, pp. 781–792.

[17] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, pp. 3861–3876, 2017.

[18] J. Umlauft, T. Beckers, M. Kimmel, and S. Hirche, "Feedback linearization using gaussian processes," in *IEEE Conference on Decision and Control*, 2017, pp. 5249–5255.

[19] Y. Lin and E. D. Sontag, "Control-lyapunov universal formulas for restricted inputs," *Control-Theory and Advanced Technology*, vol. 10, pp. 1981–2004, 1995.

[20] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle, "Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics," *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 876–891, 2014.

[21] H. Wendland, *Scattered data approximation*. Cambridge university press, 2004, vol. 17.

[22] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press, Cambridge, MA, 2006, vol. 2, no. 3.

[23] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proceedings of the 27th International Conference on Machine Learning*. Madison, WI, USA: Omnipress, 2010, p. 1015–1022.

[24] S. R. Chowdhury and A. Gopalan, "On kernelized multi-armed bandits," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 2017, p. 844–853.

[25] R. A. Horn, "The hadamard product," in *Proceedings of Symposia in Applied Mathematics*, vol. 40, 1990, pp. 87–169.

[26] M. Verhaegen and V. Verdult, *Filtering and system identification: a least squares approach*. Cambridge university press, 2007.

[27] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, vol. 30, pp. 908–918.

[28] H. Ravanbakhsh and S. Sankaranarayanan, "Learning control lyapunov functions from counterexamples and demonstrations," *Autonomous Robots*, vol. 43, no. 2, p. 275–307, Feb. 2019.